18 | BDD 是什么东西？

2021-09-13 郑晔

《程序员的测试课》 课程介绍 >

讲述：郑晔

时长 12:59 大小 11.91M

你好，我是郑晔！

在扩展篇中，我们要讨论的是在不同方向上的写测试探索。在上一讲里，我给你介绍了 TDD。TDD 是在写测试的时机上进行了不同的探索。这一讲，我们再来讲另一个实践——BDD，它是在写测试的表达方式上进行的不同探索。

我们都知道，在软件开发中最重要的一个概念就是分层，也就是在一些模型的基础上，继续构建新的一些模型。程序员最耳熟能详的分层概念就是网络的七层模型，只要一层模型成熟了，就会有人基于这个模型做延伸的思考，这样的做法在测试上也不例外。

当 JUnit 带来的自动化测试框架风潮迅速席卷了整个开发者社区，成了行业的事实标准，就开始有人基于测试框架的模型进行延伸了。各种探索中，最有影响力的就是 BDD。

行为驱动开发

BDD 的全称是 Behavior Driven Development，也就是**行为驱动开发**。BDD 这个概念是 2003 年由 Dan North 提出来的。

单元测试框架写测试的方式更多的是面向具体的实现，这种做法的层次是很低的，BDD 希望把这个思考的层次拉高。拉到什么程度呢？软件变化的源动力在业务需求上，所以，最好是能够到业务上，而校验业务的正确与否的就是业务行为。这种想法很大程度上是受到当时刚刚兴起的领域驱动设计（Domain Driven Design）中通用语言的影响。在 BDD 的话语体系中，“测试”的概念就由“行为”所代替，所以，这种做法称之为行为驱动开发。

Dan North 不仅仅提出了概念，而且为了践行他的想法，他还创造了第一个 BDD 的框架：JBehave。后来又改写出基于 Ruby 的版本 RBehave，这个项目后来被并到 RSpec 中。

好，了解了 BDD 的由来，接下来，我们就来看看采用 BDD 的方式进行开发，测试会写成什么样子。

今天最流行的 BDD 框架应该是 Cucumber，它的作者就是 RSpec 的作者之一 Aslak Hellesøy。从最开始基于 Ruby 的 BDD 框架发展成今天，Cucunber 已经变成了支持很多不同程序设计语言的 BDD 测试框架，比如常见的 Java、JavaScript、PHP 等等。

下面是一个 BDD 的示例，其场景就是我们前面实战的内容。

复制代码

```
Scenario: List todo item
  Given todo item "foo" is added
  And todo item "bar" is added
  When list todo items
  Then todo item "foo" should be contained
  And todo item "bar" should be contained
```

从这个例子我们不难看出，**BDD 的测试用例有很强的可读性**。即便我们不熟悉技术，单凭这段文字，我们也能看出这个用例想表达的含义。这也就是我们前面说 BDD 测试用例更贴

近业务的原因。它希望成为业务人员和技术团队之间沟通的桥梁，所以，它的表述方式更贴近于业务。

虽然这个表述已经很贴近业务了，但它并不是自然语言描述，而是有一种特定的格式，其实这是一门领域特定语言（Domain Specific Language，简称 DSL），称之为 Gherkin。

不要看到一门新的语言就被吓退，其实它非常简单。这里的核心点就是它的描述格式：“Given...When...Then”。**Given 表示一个假设前提，When 表示具体的操作，Then 则对应着这个用例要验证的结果。**

我们在第 5 讲谈到测试的结构时说过，测试一般包含四个阶段：准备、执行、断言和清理。把它对应到这里，Given 对应着准备，When 对应执行，而 Then 对应断言。至于清理，这个阶段会做一些资源释放的工作，不过这个工作属于实现层面的内容，在业务层面上意义不大，所以在以业务描述为主要目标的 BDD 中，这个阶段是不存在的。

了解了格式，我们再来关注具体的内容。首先，这里描述的行为都是站在业务的角度进行叙述的。其次，Given、When、Then 都是独立的，可以自由组合。这也就意味着，一旦基础框架搭好了，有人就可以使用这些基础语句来编写新的测试用例，甚至可以不需要技术人员参与。

从这里我们不难看出，Gherkin 语言本身有一个很好的目标，与其说它是为了技术人员设计的，不如说它是为了业务人员设计的。

Gherkin 语言这层只提供了业务描述，作为程序员我们很清楚，这层描述并不能直接发挥作用，必须要有一个具体的实现。那具体的实现要放在哪里呢？这就轮到**胶水层**（Glue）发挥作用了，这个将测试用例与实现联系起来的胶水层，在 Cucumber 的术语里，称之为步骤定义（Step Definition），下面就是一个步骤定义的示例。

复制代码

```
public class TodoItemStepDefinitions ... {
  private RestTemplate restTemplate;

  public TodoItemStepDefinitions() {
    ...

    Given("todo item {string} is added", (String content) ->
```

```
        addTodoItem(content)
    );

    ...
  }

  private void addTodoItem(final String content) {
    AddTodoItemRequest request = new AddTodoItemRequest(content);
    final ResponseEntity<String> entity =
            restTemplate.postForEntity("http://localhost:8080/todo-items", req
    ...
  }
}
```

既然步骤定义是 Gherkin 文件与具体实现之间的胶水，所以，理解步骤定义的关键就是知道它是如何将二者关联起来的。在这段代码中，Given 就是这样的连接点。对比一下我们就会发现， Given 里面的参数就是我们在前面 Gherkin 文件中的描述，不同的点是，这里把其中的一部分变成了参数。由此我们可以知道，**对于同样一个描述，可以根据用例的差异，采用不同的参数。**

如果说 Gherkin 语言部分几乎在各种 BDD 框架之间是通用的，那步骤定义部分则是框架强相关。这里我们采用 Cucumber Java 8 的方式进行了步骤定义，也就是采用 Given 方法进行定义，如果你去看其它的资料，也会看到基于 Annotation 的定义，这就是选择不同依赖程序库的结果。

到了具体的实现上，程序员就很有底气了。在这里我们根据业务动作进行相应的处理。在上面这段代码中，添加 Todo 项就是向自己编写的服务发出了一个 POST 请求。

这些东西理解起来都很容易，唯一需要稍微注意一点的是，给 Then 编写代码时，因为它是表示断言的，在这个部分我们一定要写出断言，比如像下面这样。

复制代码

```
Then("todo item {string} should be contained", (String content) -> {
    assertThat(Arrays.stream(responses)
            .anyMatch(item -> item.getContent().equals(content))).isTrue();
});
```

上面这段代码的更多细节实现，你可以去参考我们的实战项目。

实战中的 BDD

现在我们已经有了对 BDD 的初步了解，接下来，我们就来看看在实际的项目中可以怎样使用 BDD。

前面我们已经知道了，Gherkin 语言是面向业务人员的。不同于写代码我们只能用英文，Gherkin 在设计时就考虑到了业务人员的实际需要，所以它的设计本身是本地化的。我们甚至可以用中文编写测试用例，下面就是一个登录的测试用例。

```
复制代码
1 假定 张三是一个注册用户，其用户名密码是分别是 zhangsan 和 zspassword
2 当 在用户名输入框里输入 zhangsan，在密码输入框里输入 zspassword
3 并且 点击登录
4 那么 张三将登录成功
```

这个用例怎么样呢？或许你会说，这个用例写得挺好。如果你这么想，说明你是站在程序员的视角。我在前面已经说过了，BDD 需要站在业务的角度，而这个例子完全是站在实现的角度。如果登录方式有所调整，用户输完用户名密码自动登录，不需要点击，那这个用例是不是需要改呢？下面我换了一种方式描述，你再感受一下。

```
复制代码
1 假定 张三是一个注册用户，其用户名密码是分别是 zhangsan 和 zspassword
2 当 用户以用户名 zhangsan 和密码 zspassword 登录
3 那么 张三将登录成功
```

这是一个站在业务视角的描述，除非做业务的调整，不用用户名密码登录了，否则这个用例不需要改变。即便实现的具体方式调整了，需要改变的也是具体的步骤定义。所以，**想写好 BDD 的测试用例，关键点在用业务视角描述。**

既然 BDD 的用例更多偏向业务视角，所以在真实的项目中使用它时，我们更多偏向于把它当做验收测试的工具来用。这里就会有一个我们常常忽略的点：业务测试的模型。很多人的第一直觉是，一个测试要啥模型？

既然 BDD 更多的使用场景是复杂的验收场景，所以，相应地我们也要为测试场景进行建模。还记得我们讲好测试应该具备的属性吗？其中一点就是专业性。对于复杂场景而言，

想要写好测试同写好代码是一样的，一个好的模型是不可或缺的。

这方面一个可以作为参考的例子是做 Web 测试常用的一个模型：Page Object。它把对页面的访问封装了起来，即便你在写的是步骤定义，你也不应该在代码中直接操作 HTML 元素，而是应该访问不同的页面对象。

以前面的登录为例，我们可能会定义这样的页面对象。

复制代码

```
public class LoginPage {
    public boolean login(String name, String password) {
      ...
    }
}
```

如此一来，在步骤定义中，你就不必关心具体怎么定位到输入框会让代码的抽象程度得到提升。当然这只是一个参考，面对你自己的应用时，你要考虑构建自己的业务测试模型。

BDD 的延伸

最后，我们再来说说 BDD 的一些延伸。从上面的内容我们可以知道，BDD 的用例和普通测试的用例只是在表述方式上有所差异，从结构上看，二者几乎是完全等价的。所以，只要你想，完全可以采用 BDD 的方式进行从单元测试到系统测试所有类型的测试。

所以我们会看到，在行业里还有一些 BDD 风格的单元测试框架，其中最典型的就是 RSpec。我从 RSpec 的文档上截取了一段代码，你可以感受一下。

复制代码

```
RSpec.describe Order do
  it "sums the prices of its line items" do
    order = Order.new
    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(1.11, :USD)
    )))
    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(2.22, :USD),
      :quantity => 2
    )))
    expect(order.total).to eq(Money.new(5.55, :USD))
```

```
12    end
13  end
```

其实，它与前面的 Cucumber 用例还是有很大差异的，因为它属于单元测试的范畴，所以没有像 Gherkin 部分那种面向于业务人员的描述。但同时你也能看到，它同传统的 xUnit 框架有着很大的不同，主要是框架本身会引导你写出更具描述性的代码。

BDD 的另外一个延伸方向是对需求进行文档化的表述。既然 BDD 是在朝着业务方向靠近，争取让业务人员能够很好地理解这些测试用例，那从本质上来说，它就起到了文档的作用，这个文档和真实实现是紧密相关的，是一种“活”文档（Living Document）。活文档指的是持续更新的文档，这个概念本身不局限于技术领域。Cucumber 本身有对活文档的支持，它可以与 JIRA 去集成，可以直接把 Cucumber 测试用例变成文档。

既然要写文档，那就不局限于是否采用 BDD 这样的格式，所以，还出现了像 Concordion 这样的工具，甚至可以让我们把验收用例写成一个完整的参考文档。最开始它支持用 HTML 的方式写文档，现在也支持用 Markdown 的方式来编写文档。

无论是 BDD 也好，活文档也罢，它们背后还有一个概念，叫做实例化需求（Specification by Example，SbE），也就是用实例的方式对需求进行阐述，你可以看到 BDD 和活文档就是通过这种方式在将需求表现出来。

总之，如果你对这个方向有兴趣，前面还是有很多东西可以探索。总的来说，它就是让技术团队不再局限于技术本身，更加贴近业务，这和整个行业的发展趋势是高度吻合的。

总结时刻

这一讲，我们讲了 BDD，也就是行为驱动开发。这种思想是站在 xUnit 的框架基础之上，让测试用例的表达更贴近业务行为。

我用 Cucumber 这个今天最为流行的 BDD 框架给你介绍了如何编写测试用例，你只要记住 “Given…When…Then” 的格式，就算抓住了 Gherkin 语言表述的核心。

在实际的项目中使用 BDD 我们可以采用本地化的表述方式，不过，重点是要让测试用例贴近业务而非实现细节。一般来说，BDD 多用于验收测试，所以相应地，我们在编写步骤定义时，对于复杂业务可以考虑构建业务测试模型，对实现细节进行封装。

最后，我们还谈到了 BDD 的延伸，无论是 BDD 风格的单元测试框架，还是活文档、实例化需求，这些都是你可以进一步探索的东西。

如果今天的内容你只能记住一件事，那请记住：**技术团队要更加贴近业务。**

思考题

今天我们讨论的 BDD 更多是用在验收测试中的，你的团队是怎么做验收测试呢？欢迎在留言区分享你的经验。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 17 | TDD 就是先写测试后写代码吗？

下一篇 答疑解惑 | 那些东西怎么测？

精选留言 (2)

写留言

大碗
2021-09-13

老师的BDD项目例子里，写了一个TodoItemStepDefinitions, 然后调用了多个接口，完成了一个完整的业务功能。相当于验收了一个“需求”

展开

作者回复: 多谢补充

1

亦无
2021-09-16

如果换个角度来看，BDD 也可以认为是一种集成测试，只不过是按照业务场景进行的，目

的更明确的集成。

从另一个角度来说，BDD 和 TDD 的道理是一样的，都是让开发者，不仅仅停留在代码的层面，而是从 B 或 T 的角度进行思考，从而编写/设计出更完整的实现。

展开

1